Vol.2021-HPC-180 No.1
2021/7/20

低精度計算を活用した線形方程式に対する残差反復法の改良

寺尾 剛史[1,a)]　尾崎 克久[2]　今村 俊幸[1]

概要：本稿では，線形方程式（単一または複数右辺ベクトルを持つ連立１次方程式）に対する反復改良法について述べる．この反復改良法は，LU 分解や Cholesky 分解，QR 分解等の直接法を用いて得られた線形方程式の数値解に対して，反復的に数値解の精度を改善する手法である．浮動小数点数を用いた反復法は Moler によって提案されており，単精度計算が倍精度計算と比較して高速である場合に，混合精度計算が有効となることが Buttari らによって示されている．近年，GPU 環境において半精度や単精度等の比較的に低精度な計算を高速に処理することが可能となった．これにより，直接法において計算時間やメモリの観点からネックとなる行列の分解を低精度演算で計算し，その数値解に対して反復改良を用いて倍精度相当の解を得る混合精度計算が非常に有効となる．我々は，部分的に低精度の行列計算を適用し，収束までの反復回数を低減する手法を提案する．また，数値実験結果を用いて計算速度，解の収束に関する有効性を示す．

キーワード：線形方程式，混合精度数値計算，残差反復法，GPGPU，丸め誤差解析

1. はじめに

本稿では，単一または複数右辺ベクトルを持つ連立１次方程式

$$AX = B, \quad A \in \mathbb{R}^{n\times n}, \quad B \in \mathbb{R}^{n\times r} \qquad (1)$$

を満たす X の近似値 $\hat{X}$ を計算する数値計算法について考える．ここでは，係数行列 A に対する Doolittle の LU 分解（以後，単に LU 分解）を用いた直接計算と反復改良法を用いた混合精度数値計算法を紹介する．LU 分解とは，正則である係数行列を $PA = LU$ と分解する方法であり，L は単位下三角行列，U は上三角行列，P は置換行列である．このように行列を分解できた場合，前進・後退代入を用いて

$$X = U^{-1}L^{-1}PB$$

と X を計算することが出来る．行列 B の列数 r が比較的小さい場合（$r \ll n$），行列の分解が直接法全体の大部分のコストを占める．よって，LU 分解を半精度や単精度の低精度な演算で計算することによって高速に近似解を求めることが可能となる．一方で，LU 分解の精度は線形方程式の数値解に対する精度に大きく影響する．よって，低精度演算による行列の分解を用いた際に，反復改良が必要となる．浮動小数点演算を用いた反復改良法は 1967 年に Moler により提案されている [1]．また，1980 年には Skeel らによって，十分な精度で残差を計算した場合に数値解の精度が改善されることが解析的に示された [2]．さらに，単精度計算が倍精度計算と比較して十分に高速である場合は，混合精度計算が実用的であることが知られている [3], [4]．これらの手法は誤差解析も研究されており，様々な結果が提案されている [5], [6], [7], [8]．また，悪条件な係数行列に対応する手法も提案されている [9], [10], [11]．近年は，GPU の低精度演算を用いた手法が提案されている [12]．先行研究の多くは [2] の解析結果に基づく計算精度が用いられている．しかし，低精度計算を用いた際の数値解の改善については議論されておらず，解析もなされていない．

近年の GPU では，単精度計算が倍精度計算と比較して非常に高性能である機種が開発されており，例えば NVIDIA の Geforce RTX 3090 では，単精度計算の理論性能は倍精度計算の理論性能の約 64 倍である．一方で，残差反復には高精度な行列計算が必要であるため，GPU 上では比較的に高コストになる可能性がある．このように，現在の計算機環境の性能を十分に発揮するためには，反復改良で用いられる高精度計算の計算コストを低減する手法が必要である．これは，右辺ベクトルの数が大きい場合や反復回数が多くなる場合に重要となる．本研究では，部分的に低精度の行列計算を適用し，数値解が収束するまでの反復回数を低減する手法を提案する．

[1] 理化学研究所計算科学研究センター
[2] 芝浦工業大学システム理工学部
[a)] takeshi.terao@riken.jp

2. 反復改良法

本稿では，LU 分解を単精度で行い，得られた近似解 $\hat{X}$ を倍精度計算で得られる近似解の精度と同程度になるまで反復改良する手法を考える．線形方程式 (1) に対する近似解 $\hat{X}$ を反復改良する手法を紹介する．$\hat{X}$ の誤差を $E := X - \hat{X}$ とする．このとき，

$$E \approx A^{-1}\hat{S} \to \hat{E}, \quad \hat{S} \leftarrow B - A\hat{X} \tag{2}$$

が成り立つ．このとき，$A^{-1}\hat{S}$ の計算には初期近似解 $\hat{X}$ を計算する際に用いた係数行列の分解結果を用いることが出来る．よって，

$$X = \hat{X} + A^{-1}(B - A\hat{X}) \approx \hat{X} + \hat{E} \to \hat{X} \tag{3}$$

のように解の精度を改善する．数値計算では，誤差である E を近似的に計算するため，(3) の改良を反復的に行う必要がある．このとき，$\hat{S}$ の計算を倍精度浮動小数演算で行うことで，数値解が十分に改善することが示されている [2]．同様に，$A^{-1}\hat{S}$ の計算精度は単精度で十分である．

以下に，反復改良法のアルゴリズムを紹介する [1]．アルゴリズム中の括弧内には使用する BLAS・LAPACK のルーチン名を記載した．

Algorithm 1 (IR) 反復改良法

Data: $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times r}$.
Result: An approximate solution $\hat{X}$ for $AX = B$ using LU decomposition and forward / backward substitutions;
1: Solve $A\hat{X} = B$ using single precision (`sgetrf`);
2: **repeat**
3: $\hat{S} \leftarrow B - A\hat{X}$ using double precision (`dgemm`);
4: $\hat{S}$ is converted to single precision from double precision;
5: $\hat{E} \leftarrow A^{-1}\hat{S}$ using single precision (`sgetrs`);
6: $\hat{E}$ is converted to double precision from single precision;
7: $\hat{X} \leftarrow \hat{X} + \hat{E}$ using double precision;
8: **until** $\hat{X}$ is accurate enough;

Algorithm 1 における一反復に必要なコストは，倍精度計算が $2n^2r$ flops[*1]であり，単精度計算が $2n^2r$ flops である．つまり，LU 分解の計算量が $2n^3/3$ flops であるから，r が小さい場合の反復改良のコストは小さい．一方で，倍精度計算のみを用いる場合は係数行列 A を LU 分解の結果に上書きできる．よって，Algorithm 1 の混合精度計算の場合は，LU 分解の結果と係数行列の両方を保存する必要があるため，少なくとも 1.5 倍のメモリが必要となる．

つぎに，残差反復法における誤差解析を紹介する．ベクトル $x \in \mathbb{R}^n$，行列 $A \in \mathbb{R}^{n\times n}$ に対するノルムとして

$$\|x\|_p = \sqrt[p]{\sum_{k=1}^{n} |x_i|^p}, \quad \|A\|_p = \max_{\|x\|_p \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$$

を用いる．また A が正則の場合の条件数を

$$\kappa_p(A) = \|A\|_p \|A^{-1}\|_p$$

とする．k を反復回数とすると，以下が経験的に成り立つ．

$$\frac{\|e_i\|_2}{\|x_i\|_2} = \alpha^k, \quad \alpha = \mathcal{O}(\sqrt{n})\kappa_2(A)u_s, \quad \alpha < 1 \tag{4}$$

ただし e_i と x_i は，それぞれ E と X の i 番目の列ベクトルとし，$u_s = 2^{-24}$（単精度浮動小数点数の単位相対丸め）である．また，倍精度のみで計算を行う場合

$$\frac{\|x_i - \hat{x}_i\|_2}{\|x_i\|_2} = \mathcal{O}(\sqrt{n})\kappa_2(A)u \tag{5}$$

が期待される（$u = 2^{-53}$）．(4) より，$\mathcal{O}(\sqrt{n})\kappa_2(A) \approx u_s^{-1}$ のとき，十分な精度になるまでの反復回数が多くなる．

また，この手法では扱える係数行列の条件数が $\kappa_2(A) \lesssim 2^{24}$ 程度である．つまり，倍精度計算のみを用いた場合に扱える問題の条件数は $\kappa_2(A) \lesssim 2^{53}$ 程度であるから，完全に互換性を持つわけではない．このような問題に対応する手法として [9], [10], [11] などが提案されている．

3. 提案手法

従来手法（Alg. 1）の計算コストが大きくなる条件は

- 右辺ベクトルの数が多い，または
- 反復回数が多い（条件数が大きい）

場合である．提案手法は，上記の条件を満たす際に効率的に数値解を収束させる手法である．一方で，単一右辺ベクトルかつ反復回数が少ない場合は，反復改良に必要なコストは LU 分解のコストと比較して非常に小さいため，さらなる高速化の必要性が小さい．

本稿では，2 ステップ型の反復改良法を提案する．従来手法は $\hat{X}$ を $\hat{E}$ を用いて修正したが，提案手法では (2) の $\hat{E}$ の精度を改善する行列 $\hat{F}$ を計算する．つまり，線形方程式

$$AE = \hat{S}$$

を考える．ここで，(3) と同様に $\hat{F} \leftarrow A^{-1}(\hat{S} - A\hat{E})$ とおくと

$$E = \hat{E} + A^{-1}(\hat{S} - A\hat{E}) \approx \hat{E} + \hat{F} \to \hat{E}$$

のように，$\hat{F}$ を計算することで $\hat{E}$ の精度を改善することが出来る．また，$\hat{S} - A\hat{E}$ を倍精度計算で行う場合，$\hat{E}$ の精度が改善する [2]．本稿における主張は，$\hat{S} - A\hat{E}$ を単精度で計算した場合でも十分な $\hat{E}$ の改善が行われることである．以下にそのアルゴリズムを示す．

Algorithm 1 では，残差 $\hat{S} \leftarrow A\hat{X}$ より得られる修正項 $\hat{E}$

[*1] Floating-Point Operations の略であり，浮動小数点演算の回数を表す．簡略化のため，最高次以外の項は省略する．

Algorithm 2 (IR2) 2 ステップ型反復改良法

Data: $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times r}$.
Result: An approximate solution $\hat{X}$ for $AX = B$ using LU decomposition and forward / backward substitutions;
1: Solve $AX = B$ using single precision (`sgetrf`);
2: **repeat**
3: $\hat{S} \leftarrow B - A\hat{X}$ using double precision (`dgemm`);
4: $\hat{S}$ is converted to single precision from double precision;
5: $\hat{E} \leftarrow A^{-1}\hat{S}$ using single precision (`sgetrs`);
6: $\hat{T} \leftarrow \hat{S} - A\hat{E}$ using single precision (`sgemm`);
7: $\hat{F} \leftarrow A^{-1}\hat{T}$ using single precision (`sgetrs`);
8: $\hat{E} \leftarrow \hat{E} + \hat{F}$
9: $\hat{E}$ is converted to double precision from single precision;
10: $\hat{X} \leftarrow \hat{X} + \hat{E}$ using double precision;
11: **until** $\hat{X}$ is accurate enough;

を求めて $\hat{X} \leftarrow \hat{X} + \hat{E}$ とする．ここで，提案手法では

$$\hat{T} \leftarrow B - A(\hat{X} + \hat{E}) = B - A\hat{X} - A\hat{E} \approx \hat{S} - A\hat{E}$$

を計算する．このとき，$\hat{S} - A\hat{E}$ の計算では $B - A\hat{X}$ と比較して，桁落ちの影響が小さいために単精度計算で十分な場合がある．以降は，従来手法と同様に LU 分解の結果を用いて修正項 $\hat{F} \leftarrow A^{-1}\hat{T}$ を計算する．これにより，$\hat{E}$ の修正項である $\hat{F}$ が計算できるため，$\hat{X} \leftarrow \hat{X} + (\hat{E} + \hat{F})$ となる．$\hat{S} - A\hat{E}$ が単精度計算で計算可能な要因については付録 A.2 に記載する．

$\hat{S} - A\hat{E}$ の計算にて単精度計算で精度が十分である場合は，近似解に対して一反復で 2 回の修正を行えるため，提案手法は従来手法の約半分の反復回数で収束する．また，単精度計算が倍精度計算と比較して十分に高速である場合には $B - A\hat{X}$ が主要な計算コストになるため，単精度演算のコストを追加した際の一反復における性能の悪化も小さい．一方で，提案手法では行列 $\hat{S}$ を保存する必要があることや単精度の A が必要となるため，多くのメモリが必要となる．表 1 で，密行列に対する線形方程式の数値計算に必要なメモリ量を示す．

表 1 Comparison of cost on memory and its ratio. (Double LU: Solver using double precision LU decomposition).

Algorithm	Single	Double	Ratio ($r \ll n$)
Double LU	-	$n^2 + nr$	1
Alg. 1 (IR)	$n^2 + nr$	$n^2 + 2nr$	1.5
Alg. 2 (IR2)	$2n^2 + 2nr$	$n^2 + 2nr$	2

表 1 より，提案手法は倍精度 LU 分解を用いて解く手法と比較して少なくとも 2 倍のメモリ量を必要とする．一方で，先行研究（IR）と比較した場合のメモリ量の増加は 1.33 倍程度である．これは，行列サイズ n で換算すると約 1.15 倍程度である．よって，提案手法は多くのメモリ量を必要とするが，扱える問題の大きさへの影響は大きくはない．

4. 数値実験

ここでは，数値実験結果を紹介する．使用する数値計算環境は以下の通りである；CPU: AMD Ryzen Threadripper 3990X 64 core Processor, GPU: GeForce RTX 3090, Compiler: nvcc (cuda 11.3), Library: cuBLAS [13], cuSOLVER [14]．また，使用するテスト行列として GPU 上で実装した Higham のテスト行列 [8] を用いる．これは，指定した非負の要素を持つ対角行列 D と QR 分解を用いて得られた近似直交行列 Q_1, Q_2 からテスト行列 $A \leftarrow Q_1 D Q_2$ を求める方法である．ここで，QR 分解を cuSOLVER の cusolverDnDgeqrf と cusolverDnDorgqr，行列積を cuBLAS の cublasDgemm を用いて計算した．ここで，行列の次元を n とする．D の対角成分 d_i は $0 < d_i < d_j (i < j \leq n)$ であり，標準幾何分布に従うものとする．また，$\mathrm{cnd}(A) := d_n/d_1 \approx \kappa_2(A)$ とする．

初めに，Alg. 1, 2 で主要な行列計算となる前進後退代入 (cusolverDnSgetrs), 単精度行列積 (cublasSgemm), 倍精度行列積 (cublasDgemm) の計算性能を比較する．表 2 から，反復改良の主要となる行列計算において，提案手法は単精度計算のコストのみ増加している．つまり，単精度が高速に計算できる環境において倍精度行列積が主要なコストになる場合，提案手法の計算性能の悪化は小さくなることが期待できる．

表 2 Comparison of flops for iterative refinements (Alg. 1, 2). "Solver" indicate that abbreviation name of solver on matrix computations.

Algorithm	Solver	Single	Double
Alg. 1 (IR)	Sgetrs	$2n^2r$ flops	
	Dgemm		$2n^2r$ flops
Alg. 2 (IR2)	Sgetrs×2	$4n^2r$ flops	
	Dgemm		$2n^2r$ flops
	Sgemm	$2n^2r$ flops	

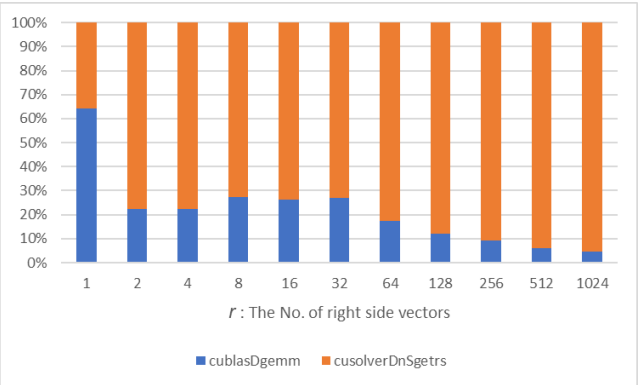

図 1 Ratio of computation time of cuSOLVER and cuBLAS in Alg. 1 ($n = 16,384$).

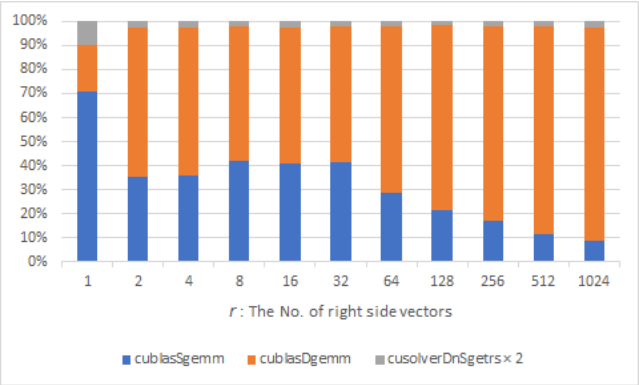

図 2 Ratio of computation time of cuSOLVER and cuBLAS in Alg. 2 ($n = 16,384$).

図 1, 2 にて，一反復における計算時間の比較結果を紹介する．これにより，$n = 1$ の場合は前進後退代入の計算が 60〜70%の計算時間を占めており，提案手法による改善は見込めない．一方で，右辺ベクトルの数が増加した場合は，倍精度行列積が計算時間の大部分を占めるため，提案手法の計算コストの増加は，相対的に小さくなる．また，$r \geq 2$ では，単精度行列積のコストは非常に小さくなる．

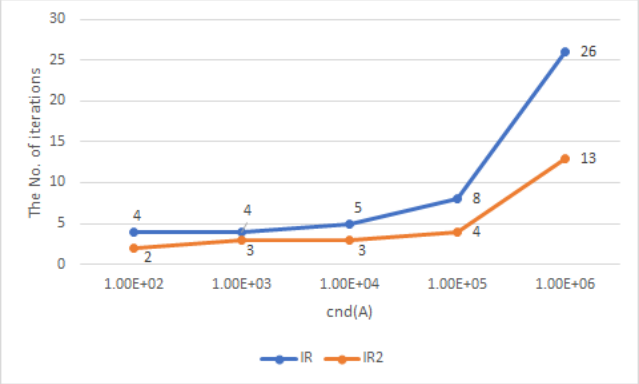

図 3 Comparison of the number of iterations ($n = 16,384, r = 1$).

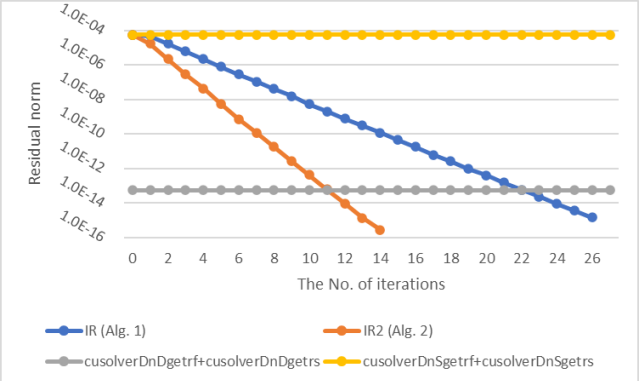

図 4 Comparison of residual norms $\|A\hat{X} - B\|_2 / \|\hat{X}\|_2$ ($n = 16,384, r = 1, \mathrm{cnd}(A) = 10^6$).

次に，図 3 では単一右辺ベクトルに対する反復回数の比較，図 4 では $\mathrm{cnd}(A) = 10^6$ での収束履歴を紹介する．また，反復の停止条件は

$$\|S\|_\infty \leq \sqrt{n}\epsilon \|A\|_\infty \|\hat{X}\|_\infty, \quad \epsilon = 2^{-53}$$

とした．以後，Alg. 1, 2 の反復停止条件はこの式を用いるものとする．図 3 より，条件数が大きくなる場合に反復回数が増加し，計算コストが大きくなることがわかる．また，提案手法は従来手法と比較して約半分の反復回数で解が収束する．つまり，LU 分解後の計算コストを考えたとき，提案手法の計算コストが従来手法の 2 倍以下であれば提案手法は有効となる．図 4 より，各反復にて改良幅が改善されている．また，反復改良では倍精度計算の LU 分解を用いた手法より良い残差が得られている．つまり，より良い反復停止条件の設定が出来れば，倍精度と同等な解をより高速に得ることが出来る．

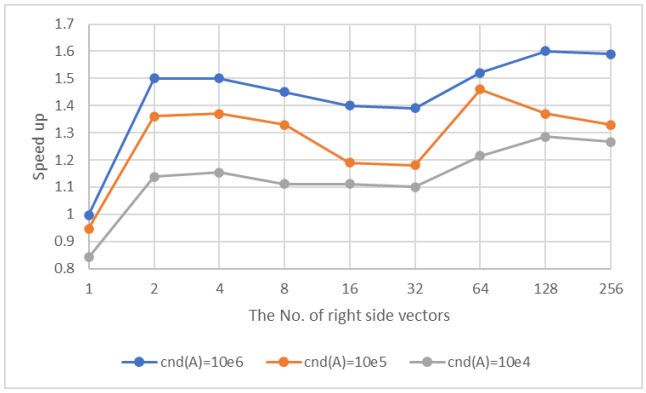

図 5 Speed up of IR2 compared to IR ($n = 16,384$).

図 5 にて，提案手法（Alg. 2）と従来手法（Alg. 1）の計算性能の比を示す．右辺ベクトルの数が大きい場合，図 5 より，条件数が比較的大きい $\mathrm{cnd}(A) = 10^6$ の場合は，1.6 倍のスピードアップができた．一方で，$\mathrm{cnd}(A) = 10^4$ の場合は，1.3 倍のスピードアップ程度である．これは，反復回数が小さいため，メモリの確保やコピー等のコストが無視できなるためと考えられる．また，単一右辺ベクトルに対しては，前進後退代入のコストが大きいため，計算性能の悪化が発生する．しかし，単一右辺ベクトルの場合は残差反復に必要なコストは LU 分解と比較して非常に小さくなることが期待できるため大きな問題とはならない．

最後に，LU 分解を含めた計算時間の比較結果を図 6 にて紹介する．この図から，複数右辺ベクトルに対して，提案手法は計算性能の改善が確認できた．また，$r = 256$ において，従来手法では混合精度計算より倍精度計算が効率的であるのに対して，提案手法は倍精度計算より高速に近似解が得られた．

5. まとめ

本論文では，単精度行列計算を用いた 2 ステップ型反復

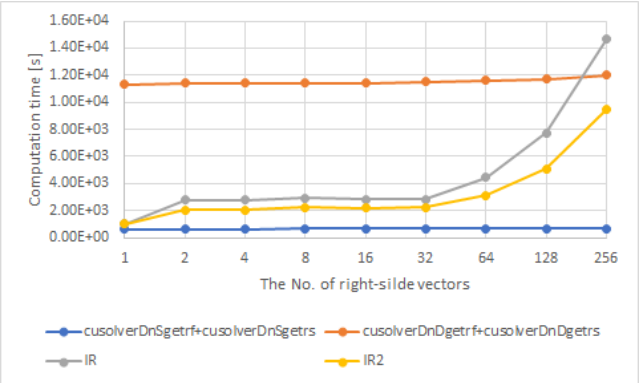

図 6 Comparison of computation times ($n = 16,384, \mathrm{cnd}(A) = 10^6$).

改良法の提案を行った．提案手法は係数行列のサイズが大きい，または右辺ベクトルが複数の場合に，GeForce RTX 3090 上での高速化を確認した．この条件は，反復改良の計算コストが大きくなる場合と一致するため，十分に実用的である．上の両条件を満たす問題にて，最大で 1.6 倍のスピードアップを確認した．また，数値実験から提案手法の改善幅は LU 分解の結果を用いた前進後退代入の性能に依存しているため，前進後退代入が単精度行列積と同等の計算性能を発揮できた場合は，最大で 2 倍のスピードアップが期待できる．

提案手法は，係数行列が疎行列の場合や悪条件問題の場合の数値計算法（例えば [10], [11], [12] など）に応用できる技術である．

謝辞 本研究は，研究教育拠点（COE）形成推進事業「高性能・高信頼性数値計算手法の研究開発・展開と人材育成」の助成を受けた．

参考文献

[1] Moler, C. B.: Iterative refinement in floating point, *Journal of the ACM (JACM)*, Vol. 14, No. 2, pp. 316–321 (1967).
[2] Skeel, R. D.: Iterative refinement implies numerical stability for Gaussian elimination, *Mathematics of computation*, Vol. 35, No. 151, pp. 817–832 (1980).
[3] Buttari, A., Dongarra, J., Langou, J., Langou, J., Luszczek, P. and Kurzak, J.: Mixed precision iterative refinement techniques for the solution of dense linear systems, *The International Journal of High Performance Computing Applications*, Vol. 21, No. 4, pp. 457–466 (2007).
[4] Buttari, A., Dongarra, J., Kurzak, J., Luszczek, P. and Tomov, S.: Using mixed precision for sparse matrix computations to enhance the performance while achieving 64-bit accuracy, *ACM Transactions on Mathematical Software (TOMS)*, Vol. 34, No. 4, pp. 1–22 (2008).
[5] Wilkinson, J. H.: *Rounding errors in algebraic processes*, Courier Corporation (1994).
[6] Stewart, G. W.: *Introduction to matrix computations*, Elsevier (1973).
[7] Demmel, J. W.: *Applied numerical linear algebra*, SIAM (1997).
[8] Higham, N. J.: *Accuracy and stability of numerical algorithms*, SIAM (2002).
[9] Oishi, S., Ogita, T., Rump, S. M. et al.: Iterative refinement for ill-conditioned linear systems, *Japan journal of industrial and applied mathematics*, Vol. 26, No. 2-3, pp. 465–476 (2009).
[10] Kobayashi, Y. and Ogita, T.: Accurate and efficient algorithm for solving ill-conditioned linear systems by preconditioning methods, *Nonlinear Theory and Its Applications, IEICE*, Vol. 7, No. 3, pp. 374–385 (2016).
[11] Carson, E. and Higham, N. J.: A new analysis of iterative refinement and its application to accurate solution of ill-conditioned sparse linear systems, *SIAM Journal on Scientific Computing*, Vol. 39, No. 6, pp. A2834–A2856 (2017).
[12] Haidar, A., Tomov, S., Dongarra, J. and Higham, N. J.: Harnessing GPU tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers, *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE, pp. 603–613 (2018).
[13] NVIDIA: cuBLAS Library, `https://developer.nvidia.com/cublas` (2021).
[14] NVIDIA: cuSOLVER Library, `https://developer.nvidia.com/cusolver` (2021).
[15] Jeannerod, C.-P. and Rump, S. M.: Improved error bounds for inner products in floating-point arithmetic, *SIAM Journal on Matrix Analysis and Applications*, Vol. 34, No. 2, pp. 338–344 (2013).
[16] Rump, S. M. and Jeannerod, C.-P.: Improved backward error bounds for LU and Cholesky factorizations, *SIAM Journal on Matrix Analysis and Applications*, Vol. 35, No. 2, pp. 684–698 (2014).

付　　録

A.1 提案手法の改良について

本稿での提案手法 Alg. 2 では，先行研究 Alg. 1 における修正項 E を単精度計算のみを用いて改良する手法を提案した．ここで，E を反復的に改善する以下のアルゴリズムが考えられる．

Algorithm 3 は，今回扱う問題や環境では十分な性能を発揮しなかったため省略した．しかし，混合精度の種類や GPU の違いによっては有用な可能性がある．特に，倍精度より高精度な解を得るときはこの手法は有効となる可能性が大きい．

A.2 提案手法の解析

A.2.1 反復法のイメージ

まず，Alg. 2 の $\hat{S}$ と $\hat{E}$ は十分な精度で計算されていると仮定する．また，$M := P^T\hat{L}\hat{U}$ とおく．このとき，線形方程式 $AE = \hat{S}$ に関して，M を右前処理行列とおくと

$$AE = \hat{S} \Longleftrightarrow AM^{-1}(Mz) = S, \quad Mz = y, \quad y^{(0)} = \hat{S}$$

のような右前処理付き反復法と考えることが出来る．つまり，$y^{(0)}$ の修正項 $v^{(0)}$ が計算できたと仮定すると

Algorithm 3 (IRk) k ステップ型反復改良法

Data: $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times r}$.
Result: Solution vectors $\hat{X}$ approximating X in $AX = B$ using LU decomposition and triangular solve;
1: Solve $AX = B$ using single precision (`sgetrf`);
2: **repeat**
3: $\hat{S} \leftarrow B - A\hat{X}$ using double precision (`dgemm`);
4: $\hat{S}$ is converted to single precision from double precision;
5: $\hat{E} \leftarrow A^{-1}\hat{S}$ using single precision (`sgetrs`);
6: **repeat**
7: $\hat{T} \leftarrow \hat{S} - A\hat{E}$ using single precision (`sgemm`);
8: $\hat{F} \leftarrow A^{-1}\hat{T}$ using single precision (`sgetrs`);
9: $\hat{E} \leftarrow \hat{E} + \hat{F}$
10: **until** $\hat{E}$ is accurate enough;
11: $\hat{E}$ is converted to double precision from single precision;
12: $\hat{X} \leftarrow \hat{X} + \hat{E}$ using double precision;
13: **until** $\hat{X}$ is accurate enough;

$$\hat{E} \leftarrow M^{-1}(y^{(0)} + v^{(0)}) = \hat{E} + M^{-1}v^{(0)}$$

となる．本稿で扱う行列の条件は $\kappa_2(A) < 10^7$ 程度の行列であるから，軸交換付き完全 LU 分解で得られる行列 M は十分な前処理行列として作用し，一回の反復で解の改善が出来る．

A.2.2 丸め誤差解析

最後に Alg. 2 の誤差解析を行う．解析の簡略化のために P を単位行列とし（$A \approx \hat{L}\hat{U}$），$A, \hat{L}, \hat{U}$ はそれぞれ正則と仮定する．また，アンダーフローは考慮しないものとし，u_s^2 以降の項は無視できるものとする．行列に対する絶対値記号は，その行列のすべての成分に絶対値をとってできた行列を意味する．行列に対する等号・不等号は，比較する行列のすべての成分に対して等号・不等号が成立する意味で使用する．文献 [15], [16] より，

$$\hat{L}\hat{U} - A = \Delta_A, \quad |\Delta_A| \le nu_s|\hat{L}||\hat{U}| \tag{A.1}$$

$$\hat{S} - A\hat{E} = \hat{T} + \Delta_T, \quad |\Delta_T| \le (n+1)u_s(|\hat{S}| + |A||\hat{E}|) \tag{A.2}$$

が成り立つ．また，$\hat{E} \leftarrow A^{-1}\hat{S}$ に関して [8], [16] より

$$\hat{S} = (\hat{L} + \Delta_L)(\hat{U} + \Delta_U)\hat{E}$$

$$\hat{E} = (\hat{U} + \Delta_U)^{-1}(\hat{L} + \Delta_L)^{-1}\hat{S} \tag{A.3}$$

$$|\Delta_L| \le nu_s|\hat{L}|, \quad |\Delta_U| \le nu_s|\hat{U}| \tag{A.4}$$

が成り立つ．ただし，$\hat{L} + \Delta_L$ と $\hat{U} + \Delta_U$ は正則であることを仮定する．また，$\|\Delta_L\|, \|\Delta_U\| \ll 1$ ならば，

$$(\hat{L} + \Delta_L)^{-1} = (\hat{L}(I + \hat{L}^{-1}\Delta_L))^{-1} = (I + \hat{L}^{-1}\Delta_L)^{-1}\hat{L}^{-1}$$
$$\approx (I - \hat{L}^{-1}\Delta_L)\hat{L}^{-1} = \hat{L}^{-1} - \hat{L}^{-1}\Delta_L\hat{L}^{-1}$$
$$(\hat{U} + \Delta_U)^{-1} \approx \hat{U}^{-1} - \hat{U}^{-1}\Delta_U\hat{U}^{-1}$$

が成り立つ．これらの式を (A.3) に代入して

$$\begin{aligned}\hat{E} &\approx (\hat{U}^{-1} - \hat{U}^{-1}\Delta_U\hat{U}^{-1})(\hat{L}^{-1} - \hat{L}^{-1}\Delta_L\hat{L}^{-1})\hat{S}\\ &\approx \hat{U}^{-1}\hat{L}^{-1}\hat{S} - (\hat{U}^{-1}\hat{L}^{-1}\Delta_L\hat{L}^{-1} + \hat{U}^{-1}\Delta_U\hat{U}^{-1}\hat{L}^{-1})\hat{S}\\ &= \hat{U}^{-1}\hat{L}^{-1}\hat{S} - \hat{U}^{-1}\hat{L}^{-1}(\Delta_L\hat{U} + \hat{L}\Delta_U)\hat{U}^{-1}\hat{L}^{-1}\hat{S}\\ &= \hat{U}^{-1}\hat{L}^{-1}(I - (\Delta_L\hat{U} + \hat{L}\Delta_U)\hat{U}^{-1}\hat{L}^{-1})\hat{S}\end{aligned}$$

を得る．また (A.1) より

$$\begin{aligned}\hat{E} &\approx (\hat{L}\hat{U})^{-1}(I - (\Delta_L\hat{U} + \hat{L}\Delta_U)\hat{U}^{-1}\hat{L}^{-1})\hat{S}\\ &\approx (A + \Delta_A)^{-1}(I - (\Delta_L\hat{U} + \hat{L}\Delta_U)\hat{U}^{-1}\hat{L}^{-1})\hat{S}\end{aligned} \tag{A.5}$$

が成り立つ．よって，行列積 $A\hat{E}$ に対して

$$\begin{aligned}A\hat{E} &\approx A(A + \Delta_A)^{-1}(I - (\Delta_L\hat{U} + \hat{L}\Delta_U)\hat{U}^{-1}\hat{L}^{-1})\hat{S}\\ &= ((A + \Delta_A)A^{-1})^{-1}(I - (\Delta_L\hat{U} + \hat{L}\Delta_U)\hat{U}^{-1}\hat{L}^{-1})\hat{S}\\ &= (I + \Delta_A A^{-1})^{-1}(I - (\Delta_L\hat{U} + \hat{L}\Delta_U)\hat{U}^{-1}\hat{L}^{-1})\hat{S}\\ &\approx (I - \Delta_A A^{-1})(I - (\Delta_L\hat{U} + \hat{L}\Delta_U)\hat{U}^{-1}\hat{L}^{-1})\hat{S}\\ &\approx \hat{S} - (\Delta_A A^{-1} + (\Delta_L\hat{U} + \hat{L}\Delta_U)\hat{U}^{-1}\hat{L}^{-1})\hat{S}\end{aligned} \tag{A.6}$$

が成り立つ．また，(A.1) と (A.4) から

$$\|\Delta_A A^{-1}\|_\infty \lesssim nu_s\kappa_\infty(\hat{L})\kappa_\infty(\hat{U}) \tag{A.7}$$

$$\|(\Delta_L\hat{U} + \hat{L}\Delta_U)\hat{U}^{-1}\hat{L}^{-1}\|_\infty \lesssim 2nu_s\kappa_\infty(\hat{L})\kappa_\infty(\hat{U}) \tag{A.8}$$

である．よって，(A.6)，(A.7)，(A.8) から

$$\hat{S} - A\hat{E} \approx \Delta_1 \cdot \hat{S}, \quad \|\Delta_1\|_\infty \lesssim 3nu_s\kappa_\infty(\hat{L})\kappa_\infty(\hat{U}) \tag{A.9}$$

である．次に，(A.2) の $(n+1)u_s|A||\hat{E}|$ について (A.5) から

$$\begin{aligned}(n+1)u_s|A||\hat{E}| &\lesssim (n+1)u_s|A||A^{-1}||\hat{S}|\\ &\approx \Delta_2 \cdot |\hat{S}|, \quad \|\Delta_2\|_\infty \lesssim (n+1)u_s\kappa_\infty(A)\end{aligned} \tag{A.10}$$

である．(A.10) より $(n+1)u_s\kappa_\infty(A) < 1$ であるから

$$|\Delta_T| \le (n+1)u_s(|\hat{S}| + |A||\hat{E}|) \approx (n+1)u_s(I + \Delta_2)|\hat{S}|$$
$$\|\Delta_T\|_\infty \lesssim (n+1)u_s(1 + \|\Delta_2\|_\infty)\|\hat{S}\|_\infty \approx (n+1)u_s\|\hat{S}\|_\infty$$

である．ここで，この行列積で発生する丸め誤差の上限 $\Delta_2 \cdot \hat{S}$ は単精度計算であっても無視できる程度である．よって，(A.2) に (A.9) を代入し

$$\hat{T} = (\hat{S} - A\hat{E}) + \Delta_T \approx \Delta_1 \cdot \hat{S} + \Delta_T,$$
$$\|\hat{T}\|_\infty \lesssim (\|\Delta_1\|_\infty + (n+1)u_s)\|\hat{S}\|_\infty$$

が成り立つ．ここで，(A.9) から $\Delta_1 \cdot \hat{S}$ は $\hat{S} - A\hat{E}$ の真の値の近似であるため，$(n+1)u_s \ll \|\Delta_1\|_\infty$ ならば，丸め誤差の影響は小さいことが期待できる．